Aware Home Visual Perception (Part I): Design and algorithms

Zhonghao Yang Aaron Bobick

Graphics, Visualization and Usability Center
Georgia Institute of Technology, Atlanta, GA, 30332, USA
{*yangzh, afb@cc.gatech.edu*}

Abstract

In this paper we present the design details of the visual perception system in Aware Home. This paper is intend to (a) detail our design details; (b) provide insights on how different parts of the tracking system are inter-related to solve complex perception tasks; (c) document ideas and potential research directions; (d) provide performance evaluation. This work will primarily focus on algorithm and design issues, while another Technical Report will be authored to address coding issues related to this system (APIs, class structures, etc).

1 Overview

1.1 Task and settings

We are working in Aware Home Research Initiative, a smart environment with overhead cameras mounted for every kitchen, living room and hallway. Our task is to provide house-wide visual perception capability for various location-aware user-interactions.

Obviously this task can not be achieved with any single machine given current mainstream CPU capacity. For this reason, distributed perception infrastructure has been established (we will detail this in another Technical Report), which essentially provide clear APIs for synchronized communication of data beyond machine and process boundaries. Built on this infrastructure, we would like to achieve a visual perception system that can:

1. Reliably track targets (presumably people), maintain their identities across multiple rooms throughout their lifetime, under reasonably wide range of system noise (for example, illuminations disturbance);

2. Function robustly in real-time with distributed computing facilities. The tracker should not be subject to specific camera deployments / machine configurations;

3. High degree of automation or limited manual intervention during extended period of time. The service is preferably up and running 24-by-7.

The visual tracking is still an open research topic despite enormous research efforts and vast amount of literatures in the recent two decades, especially for large-area and/or multi-camera cases. The final solution can be quite different depending on specific application requirements, constraints, and other considerations. Our discussion will be concentrated on a real-time visual tracker for in-door scenarios with multiple cameras. We hope our endeavor can be as general as possible so that this system design can be helpful for other application scenarios.

1.2 Previous work

The W4 system [12] and its successor W4S [13] (with additional stereo capability) are designed for outdoor surveillance and activity analysis with monochromatic and stationary video cameras. Their targets have to be isolated, upright, and un-occluded in order to proceed. The VSAM project [6] [7] is another system focus on large-scale multi-camera, multi-target visual tracking for military awareness in an outdoor settings. However, the project's emphasis is to providing an unified interface for a human operator rather than full-automatic visual tracker. For this reason, their design is quite different from what we intend to implement here. Recently we found Tim Ellis of City University London also has good work on multi-camera visual tracking in city environment, especially their automatic scene learning technique, readers are encouraged to refer to his website for publications and ideas.

The major difference between outdoor and indoor tracking is that we can no longer safely assume targets are reasonably separated, and visual occlusions are no longer uncommon. The small distance between camera and target sometime causes significant perspective distortion, and visual modelling becomes prohibitively complicated. A number of previous work on multi-camera, multi-target visual

tracking in indoor/smart environments is reported:

The Cai's work [3] has the similar architecture as our work and uses multiple features (location, color) to facilitate consistent tracking, whereas they do not present a solution for occlusion. In a typical smart room environment, Microsoft's Krumn [23] takes advantage of multiple stereo camera modules to simplify tracking task and their system can deal with 2 to 3 people in one room in real-time. They also use color histogram to maintain identity across frames. Another interesting work is done by Huang, et. al. [16] for an intelligent room using regular and omnidirectional cameras. They can track up to 4 people in a room with some degree of success. M2Tracker [27] is most similar to our work in terms of a multi-camera and multi-target visual tracker. They explicitly use human color models to further segment blobs into individual target, and associate targets from multiple views with a region-based stereo algorithm. One potential issue is that the tracker might get confused when two targets are wearing similar clothes, especially during target initialization. Our approach simplifies the system design and improves robustness by: 1) eliminating occlusion early during feature fusing phase; 2) associating targets with more robust and cheaper geometrical constraints (compared with stereo). In the long-run, we want our system to be able to deal with multiple room and ideally deal with 5-10 people in real-time. We have not found systems elsewhere that can provide similar capabilities so far, to our best knowledge.

For system development, OpenCV and LTI package (http://ltilib.sourceforge.net) provide valuable source of robust computer vision algorithms. The latter also has nice design for large scale architecture that enables flexible algorithm development and cooperative vision modules.

For the rest of this paper, I will cover major parts of the proposed visual tracker, namely background subtraction, visual fusing, motion correspondence and labeling handover. Besides above-mentioned algorithm issues, we will also discuss the enabling middleware that glues individual algorithm modules. Special attention is given to illustrate how ambiguities are eliminated gradually so that reasonable tracking performance can be achieved as a whole. In the end, performance evaluation and discussions will be presented.

1.3 System architecture

From system engineering point of view, there exists various competing middleware designs and implementations (CORBA, DCOM, JMI, etc.) to enable distributed computing practice. Given algorithms running in separate process space or even different machines, we would like to **glue** the modules together to form a distributed yet coherent system. Here in the system we base our system on Microsoft COM/DCOM. It turned out to provide reasonable performance in general.

Generally a good design practice is to use a thin layer with well-defined interfaces and clear semantics. Within each module, we concentrate on the platform-independent algorithm design while treating the communication function as a black box. For the communication layer based on COM/DCOM, we carefully keep them separate from the algorithm code and only this portion of code are system-specific. This makes further port to other distributed computing technology (for example, grid) much more easier.

At system level, we designed the following hierarchy (Fig. 1) to illustrate both visual information abstraction and module responsibilities. The idea is that we hope to eliminate ambiguities gradually at various stages of the system. This modularization practice makes the system very robust and easy to port if needed.

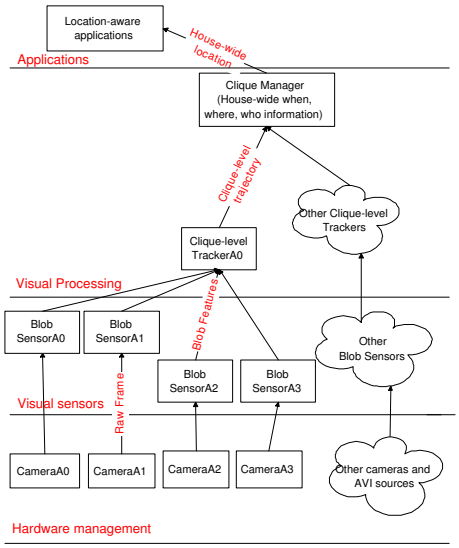

Figure 1: system schema

There will be another Technical Report detailing software engineering efforts for the system.

2 Background subtraction

Background subtraction is a well-studied however not yet fully-solved topic due to its own inherent ambiguities. The Wallflower [37] and Wang's work [42] provide quite detailed survey and we won't repeat here.

Background subtraction, motion differencing and optical flow are somewhat standard approaches for visual sensing front-ends. Optical flow computation tends to be computationally expensive. For in-door static camera settings, well-engineered background subtraction algorithm often constitutes suitable solution. Motion differencing is generally

used for outdoor settings, for example, [6] [7] switch between background subtraction and motion differencing for blob sensing for PTZ cameras since background is hard to obtain and maintain. The major issue for all above motion-feature-based algorithms is that the motion features does not necessarily correspond to the objects we are interested in tracking, a fundamental flaw that leaks various ambiguities into the rest parts of the visual tracker.

We use the algorithm proposed by [14] because it can deliver reasonable performance under reasonable wide illumination variations. The algorithm in [20] can also be used alternatively and we have an implementation in our system.

In our specific settings, we conclude that:

Deposited objects will initially appear as detected blobs since they are not registered with the background model. The gradual merging (compensation) of foreground into background provide a remedy to this problem while introducing new issue of its own: the non-moving targets will gradually cease to be detected as blobs. However, this compensation feature is essential in our working environment to accommodate depositing objects, since otherwise it will be quite awkward for real environment. We keep this feature enabled in our system.

Deposited objects and background holes will be recorded as targets when their blobs are detected. This is acceptable as long as these kinds of targets can be graduated eroded and do not interfere with other targets under tracking. Being a generic tracker, we have no knowledge and no intent to distinguish them from true people targets.

3 Visual fusing

Single camera can only provide limited field of view and fixed viewing angle, and quite often surveillance applications need multiple cameras to provide larger coverage, flexible settings and possibly different image capabilities. Visual fusing is our approach to address the issue of multi-camera integration.

The original work [44] and our extension [45] has been implemented in our visual tracker. After a period of performance evaluation, we identified some issues and will summarize the improved version here.

3.1 The design

The general idea is to exploit the overlapping of camera views, and use both temporal and spatial constraints to assist visual tracking. Given relatively high sampling frequency (for example, above 5Hz), targets tend to be near the location where they were in the previous frames. On the other hand, it's possible to cross-check targets using overlapping views if available. It's interesting to compare our work with stereo-based tracking such as [1], [24], [28], where the overlapping of camera views is exploited in stereo methodology. Our advantage is that the camera configuration is not so strictly enforced as in the stereo case, and is not computationally expensive since we do not need pixel-wise matching as in stereo. Actually this visual fusing algorithm can run comfortably in real-time and consumes little resources.

The major entities in our algorithm is polygons. Each polygon represents a set of floor grids presumably containing one or more targets of interest. Currently we discretize the floor into grids with displacement of $2cm$, a compromise between accuracy and computing requirement. Each polygon P has both lower bound constraint l_P and upper bound constraint u_P to encode ambiguity for target counting. Accurately counting targets and recording their states for every frame is impossible, due to noise and complex target interactions. For this reason, we instead record their lower and upper bound and hope eventually this will converge to the true value if proper updating is performed. Once the lower bound has converged, hysteresis can be used to stabilize output in case of small disturbances.

Without prior knowledge, polygons are initialized as:

$$l_P = 0 \tag{1}$$

$$u_P = size(P)/T_{minSize} \tag{2}$$

Equivalence notation is:

$$\{P\} = \{l_P : u_P\} = \{1 : size(P)/T_{minSize}\} \tag{3}$$

The confidence of a polygon indicates to what extent we confirm targets within the polygon, and it can be accumulated or deteriorated depending on observation. Notice the initialization of polygon Eq. 1 is different from original work by [44] due to our confidence accumulation design. u_P tends to be loose because it's often hard to provide accurate $T_{minSize}$, and we assume that targets fill every grid in the polygon like water.

In addition to bound constraints and confidence, each polygon P maintains a unordered list of targets it believes under its coverage, the length of the list is always l_P as l_P will eventually converge to the actual target count.

Targets and polygons are different in the sense that one polygon might contain multiple targets under complex interactions due to merging and split: even the use of multiple camera cannot guarantee to segment targets perfectly (consider the case there are 30 people in one room). A target T contains states about an object of interest. Currently the states include target label (targetID), location and velocity in world coordinates, size and optional color model. During target initialization, location (with the help of proper camera model), size can be computed. If we need optional color model, a implicit assumption is that targets enter the

scene individually to enable proper color model initialization. Additional information such as label, velocity will be computed in the following motion correspondence step. With proper label and tracking history, individual track can thus implicitly formed.

We use a tree structure to encode and enforce spatial and temporal constraints. Each tree node is a **polygon**, either comes from current frame (namely **P-polygons**) or is left from previous frames (namely **Q-polygons**) due to unsolved ambiguities.

For any tree node i, there are 4 properties this tree structure should always observe to ensure counting correctness:

$$l_i = \max\{l_i, \sum_{\forall j \in children(i)} l_j\} \tag{4}$$

$$l_i = \max\{l_i, l_{parent(i)} - \sum_{\forall j \in siblings(i)} u_j\} \tag{5}$$

$$u_i = \min\{u_i, \sum_{\forall j \in children(i)} u_j\} \tag{6}$$

$$u_i = \min\{u_i, u_{parent(i)} - \sum_{\forall j \in siblings(i)} l_j\} \tag{7}$$

The tree structure is updated every frame with P-polygons where spatial and temporal constraints are enforced. After the update, observations of targets as well as target count can be obtained to feed into motion correspondence algorithm. Several key steps in the updating algorithm are described here.

3.1.1 Add P-polygons

P is initialized by Eq. 1 and Eq. 2. Initially each P-polygon P acts as a single node, i.e., it does not have a parent node.

If P is found to intersect only one Q-polygon Q, simply add P as child of Q.

If P intersects exactly two Q-polygons Q_1 and Q_2, merging is detected: targets in either Q_1 or Q_2 might have moved into P. Adding P as a child to both Q_1 and Q_2 would have created an undesirable cycle in the tree. Instead, we update the tree by:

1. Locate polygon N: the closest common ancestor of Q_1 and Q_2 in the tree. Intuitively this is the smallest area that can completely cover both Q_1 and Q_2 (which in turn cover P in the new frame);

2. Create a compound Q-polygon $Q_{(1,2)}$ with the bounds initialized by the combined bounds of Q_1 and Q_2, $\{Q_{(1,2)}\} = \{l_{Q_1} + l_{Q_2} : u_{Q_1} + u_{Q_2}\}$, and target list initialized by simple concatenation of Q_1 and Q_2 (an unordered list);

3. To ensure overall counting correctness, for any node Q_k along the path from Q_1 to Q_2 via N (except N) in the tree, let l_k decreased by u_P to accommodate the worse case that at most u_P targets might have exited from polygon Q_k;

4. Add the compound Q-polygon $Q_{(1,2)}$ as the child of N; remove Q-polygon Q_1 and Q_2; add P, Q_1 and Q_2 as children of $Q_{(1,2)}$.

If P interects $\{Q_1, Q_2, ..., Q_k\}$, handle this by several nested calls. See [44] for details.

For any P-polygons P that does not overlap with any Q-polygons, consider it as a new polygon and add it directly as the child of the root.

After all P-polygons has been added into the tree, any Q-polygon Q which has multiple P-polygons as children is detected as split, and any compound Q-polygon indicates a merge occurrence.

3.1.2 Remove redundant

The redundancy removal is the same as in [44]: any Q-polygons without children is simply obsolete and removed from the tree. Intuitively, this is because ambiguities for those Q-polygons has been successfully removed, and we do not need them any more. Any Q-polygons with only one child is also obsolete and removed. In this case, the only child will be updated by the tighter between the Q-polygon and itself. Target list is also updated accordingly.

3.1.3 Update bounds

General, update bounds involves two sweeps: propagate constraints from leaves (local count) to root (global target count) by Eq. 4 and Eq. 6; propagate constraints from root to leaves by Eq. 5 and Eq. 7. Note the downward updating has to be performed in parallel to ensure correctness.

The lower bounds for P-polygons are initialized as 0 simply due to insufficient confidence. As contrast, lower bounds for Q-polygons tend to be more reliable because they incorporate and filter history information.

We illustrate the process in Fig. 2 given an imaginary scenario that P-polygon P_0 intersects with Q_0, and P_1 intersects with both Q_1 and Q_2. Numbers changed from previous step are marked italic and underline to facilitate understanding. Keep in mind that the number of targets for each polygon is always its lbc.

3.1.4 Gather observations

In order to get target observations, we traverse the tree in pre-order, i.e., visiting the node itself before visit all its children. There is nothing to do with a P-polygon but simple report all its target's locations. When visiting a Q-polygon

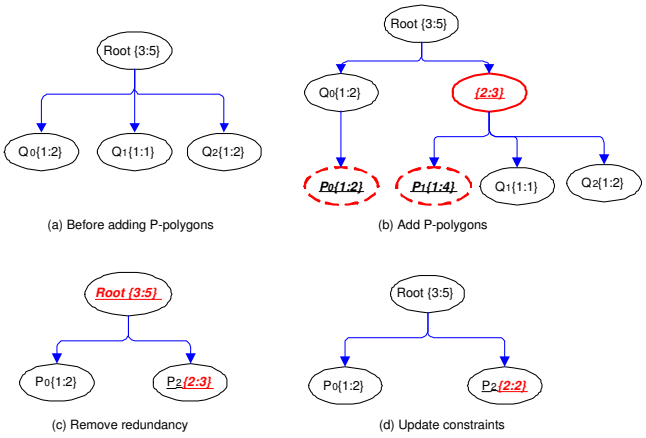

Figure 2: updating of the polygon tree

Q_i, we examine whether $\sum_{\forall j \in children(i)} l_j = l_i$. If the Q-polygon is *tight*, all targets associated with Q_i can be reliably observed in current frame (its child P-polygons), and we deliver its children as the latest observations; otherwise, to maintain temporal smooth, we still use this Q-polygons's state as output (optionally disturbed by some noise to model dynamics).

3.2 Confidence accumulation

Each polygon is associated with a confidence to measure the probability that the polygon contains confirmed targets, and we use that to design a soft decision schema on target counting and localization.

The confidence of a polygon P is mathematically modeled as a function of both P and time t:

$$p(P,t) = 1 - e^{-t/T_1} \tag{8}$$

where $T_1 = T_1(x,y)$ is the time constants depending on camera coverage of local grid. The more camera coverage, the less time constant T_1, implying a quick accumulation rate since we tend to trust observation which has visual support from more cameras.

We update confidence with Eq. (9).

$$p(P,t) = 1 - (1 - p(P,t-1)) \times e^{1/T_1(x(t),y(t))} \tag{9}$$

In the frames where a previous Q-polygon cease to be observed, we use a pseudo P-polygon, and its confidence drops exponentially:

$$p(P,t) = p(P,t-1) \times e^{1/T_2(x(t-1),y(t-1))} \tag{10}$$

where $T_2 = T_2(x,y)$ is the time constants depending on camera coverage of local grid. The less camera coverage, the longer the time constant T_2 because of unreliable observations.

Heuristically, we set both $T_1(x,y)$ and $T_2(x,y)$ to be 2, 4, 8, 16 for floor grids where 4, 3, 2, 1 camera(s) are available. $T_1(x,y)$ and $T_2(x,y)$ are set to infinite where no camera coverage is available.

Before a polygon P's confidence has reached a predefined T_{thres}, we initialize $l_P = 0$. The unconfirmed polygon P will continue to be in the tree as long as it is continuously observable. A polygon P is said to be confirmed by setting $l_P = 1$ when its confidence has accumulated above T_{thres}. Note this is only a lower-bound for the actual target count within the polygon P, and the tree update will gradually ensure the correctness of the global count. When a confirmed polygon fails to be visible in one frame, we manually substitute with a pseudo-polygon with its confidence exponentially deteriorated by Eq. 10.

When a confirmed polygon P's confidence drops below T_{thres}, we have to exclude it from global target counting by resetting $l_P = 0$ and remove it from the tree. Fig. 3 illustrates the accumulation and deterioration of confidence for polygons. T_{thres} is set to 0.5 for current system.

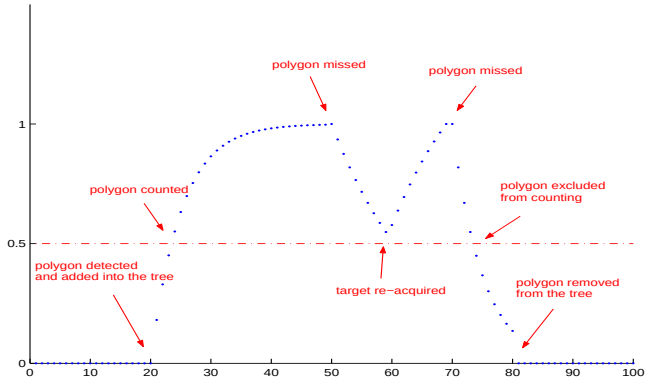

Figure 3: change of confidence for polygons

For optimization reason, we pre-compute the quantity $e^{-1/T_1(x,y)}$ and $e^{-1/T_2(x,y)}$ for each floor grid so that later confidence update will be a simply look-up operation plus a few simply arithmetic operations.

Another insight into real-world visual tracking is to explicitly model the scene to maximize tracker's robustness. Generally the better the tracker knows the world, the better performance can be achieved. However, we have to be careful not to constraint the tracker too much to a specific scene settings since that will hurt portability.

We model the floor by explicitly telling the system where the wall and furniture are. By setting $T_1(x,y)$ at those locations explicitly to infinite, we essentially dictate that new target will never originated from those grids and target will

never disappear from those grids. Example of such places includes walls, and fixed furniture.

3.3 Algorithm

Here is the algorithm for the visual fusing:

1. Model the scene by pre-compute time constants: compute $T_1(x, y)$ and $T_2(x, y)$ for each grid on the floor;
2. Get blobs from each blob detector. This fusing algorithm need signal synchronization and some additional sync code is needed;
3. With appropriate camera model information, project blobs (from multiple blob detectors) onto floor;
4. Compute the intersection of all blob projections, initialize a list of P-polygons as new observations;
5. **Add P-polygons**;
6. **Remove redundant** and **Update bounds** for the tree;
7. **Gather observations** and feed into motion correspondence for labeling assignment;
8. Iterate to next frame.

It's noted that the fusing algorithm need synchronization for each blob detectors to ensure proper functioning, see implementation details in the second part of documentation for this project.

4 Motion correspondence

4.1 The design

It has to be noted that the output of visual fusing is merely a set of target states without proper labels assigned. Motion correspondence is introduced to help form temporally consistent tracks. Thus the clique-level visual tracker is composed of visual fusing and motion correspondence. This separation of functionalities within visual tracker can be considered as a divide-and-conquer effort that attempts to simplify design for individual modules and provide better insights into understanding their respective strength and limitations.

Motion correspondence algorithms are widely studied in various domains such as motion analysis, stereo, optical flow and structure from motion. Originally for point-like or identical features, the algorithms are often with domain-specific constraints. Under strong assumptions as illustrated in our previous work [45], even the simple nearest-neighbor assignment can yield reasonable performance. However, in order to have a solution that works under minimal assumptions, we have the following considerations:

The target count is unknown a prior and changing since people enter/exit the scene from time to time. The inclusion for varying number of targets greatly complicates the problem since it's inherently indistinguishable with sensor ambiguities, and thus conflicting requirements. To make this clear, it's just hard to tell the difference between a target recovery after long-time disappearance and a new target appearance. It's still not clear to us how much additional benefit can be achieved with the use of additional color models.

Sensors might fail to detect targets due to temporary weakness of visual signals, illumination change or even its own failure. The possibility of false negative is not neglectable and needs to be modeled in proper way. As contrast, we fortunately have overlapping views to enable reliable cross-checking, which means false positive is minimal: once observation is detected, there has to be something out there. This fact turns out to be very useful and makes the whole complexity under control.

In our application, target labeling could be much easier with the help of distinctive and accurate color models. However, it's not always we can obtain accurate color model due to camera limitation of field of view, and more study need to be done to determine how to combine color feature from multiple views. In case where people wearing similar clothes, the color model simply lost its ability to distinguish targets. Ideally we would built color and size features in addition to positional ones, the target labeling problem will downgrade gracefully to positional-based motion correspondence in case of color and size ambiguities. This is our current work focus to study how to dynamically and adaptively change matching feature to achieve robust motion correspondence, similar to the idea proposed in [17] [18].

4.2 Basic concepts

We've developed techniques based on Veenman et. al's work [39] [40] in our system. In essence, their approach relies on minimization cost functions.

First we formulate the problem: suppose for the current frame t, we get m_t *original measurements*. One measurement contains position, and optional size, color model for one target. The terminology is to distinguish from *dummy measurements* as introduced later. Our task is to associate them with M *existing tracks* up to now, considering the possibility of dynamic target enter/exit and detector failure.

Hungarian algorithm (also referred as Munkres' Assignment Algorithm) is a polynomial runtime complexity algorithm for solving the minimal cost assignment problem. We use one adaptation found on the Internet (http://www.public.iastate.edu/ ddoty/HungarianAlgorithm.html)

in our implementation.

We denote original cost matrix $C^t = \{c_{i,j}\}_{(0 \le i < M, 0 \le j < m_t)}$ (notice the zero-based indexing). In order to handle detector failure, we expand the cost matrix by 1) propose m_t *new tracks* to accommodate the worst case that all m_t *original measurements* had originated from m_t new tracks; 2) propose M *dummy measurements* to account for the worst case that the detector fails to detect all *existing tracks* and thus each *existing track* has produced one *dummy measurement*. This will make a square cost matrix of $C^t_\lambda = \{c_{i,j}\}_{(0 \le i < M+m_t, 0 \le j < M+m_t)}$, with each $c_{i,j}$ representing the assignment cost between ith track with jth measurement.

Inherited from [39], ϕ_{thres} is the cost threshold to tell whether it's necessary to start a new track instead of associating measurement with a existing track. This value depends on application requirements and system noise level, and actually contributes a lot to the final assignment performance. ϕ_{inf}, an arbitrary large value, is an approximate for infinite. If we confine (normalize) cost function to be within the range of 0 to 1, and $0 \le \phi_{thres} \le 1$, ϕ_{inf} can be set to $1 + \epsilon$.

There is an internal termination counter for each *existing track* to count how many successive frames the track has been assigned with *dummy measurements*. The counter is reset when the track gets one *original measurement*. Target termination is triggered when the counter exceeds N_{term}. The inherent reason for such design is that *dummy measurements* can be caused by either temporary detector failure or permanent target exit, which, can not be reliable decided at the first frame of occurrence. The practical strategy is to postpone several frames to see if the ambiguity can be solved. The heuristic N_{term} is to delay such decision making.

4.3 The algorithm

Here is the motion correspondence algorithm:

1. Synchronize blob sensing channels, and produce a set of target measurements by visual fusing;

2. Compute each cost $c_{i,j}$ in the extended square cost matrix C^t_λ as following:

   (a) existing tracks to original measurements, i.e., $0 \le i < M$, $0 \le j < m_t$, compute $c^t_{i,j}$ depending on ith track's state with jth measurement. We use our own cost function (explained in the next section). If any of the preset hard constraints are violated, set $c^t_{i,j} = \phi_{inf}$ to invalidate any possible assignment;

   (b) new tracks to original measurements, i.e., $M \le i < M+m_t$, $0 \le j < m_t$, set $c^t_{i,j} = \phi_{thres}$. This implies each *original measurement* has to check with every *existing track* for possible smaller cost before associating with one *new track*;

   (c) existing tracks to dummy measurements, i.e.,$0 \le i < M$, $m_t \le j < m_t + M$, set $c^t_{i,j} = \phi_{thres}$. This implies each *existing track* has to check every possible *original measurement* before giving up and associating with one *dummy measurement*;

   (d) new tracks to dummy measurements, i.e., $M \le i < M + m_t$, $m_t \le j < m_t + M$, set $c^t_{i,j} = \phi_{thres}$. This cost will discourage any possible association between them;

3. Apply the Hungarian algorithm to this extended cost matrix which results in the minimal cost assignment;

4. For each assignment:

   (a) One *existing track* to one *original measurement*: the target is detected and updated directly, with termination counter reset;

   (b) One *new track* to one *original measurement*: one new target is detected and initialized by the measurement;

   (c) One *existing track* to one *dummy measurement*: increase the termination counter. If the termination counter reading is great than N_{term}, terminate this *existing track* due to permanent target exit; otherwise extrapolate position/velocity from track history;

   (d) One *new track* to one *dummy measurement* is simply discarded.

5. Sent tracks out to global location manager, go back to step 1 for next iteration of frame data;

We have the freedom to preset hard constraints such as the maximum velocity depending on domain knowledge.

To understand more about what happened in the optimization matching, let's assume $M \le m_t$, since there is no false positive based on our assumptions and thus we fully trust measurements, at least $m_t - M$ *new tracks* have to be detected and created. If one *existing track* can not be associated with any *original measurement* due to temporal detector failure or target exit, an additional *new track* is created to account for the available *original measurement*.

Similarly, when $M \ge m_t$, at least $M - m_t$ *dummy measurements* has to be reported. If one *original measurement* can not have assignment cost small than ϕ_{thres} for any *existing track*, there must be one more *dummy measurement*, and one *new track*.

To sum up, detector failure is handled by keeping the *existing track* for at most N_{term} frames; after which the *existing track* is removed to account for permanent target exit. *New track* is initiated when there is no *existing track* to reasonably claim for one *original measurement* with similarity small than ϕ_{thres}.

4.4 Cost functions

To model regular cost function $c_{i,j}^t, 0 \leq i < M, 0 \leq j < m_t$, we initially tried the individual motion model im2, as originally introduced by [33] and explained in [39]. The major issue, however, is that people are not missiles, where where dynamics are well-defined. People can walk/stop very abruptly, and this lack of smooth motion approximation often break the system. We turned to a simpler and heuristic combination of both location and velocity score, with each score being truncated square distance. Each score is a function of ith track (T_i) and jth measurement (M_j). D_{cutoff} is a cut-off distance of $100cm$ in our system.

$$f_{loc}(T_i, M_j) = \begin{cases} (\frac{\|D_{loc}\|}{D_{cutoff}})^2 & \text{if } \|D_{loc}\| \leq D_{cutoff}; \\ \phi_{inf} & \text{if } otherwise. \end{cases} \tag{11}$$

$$f_{vel}(T_i, M_j) = \begin{cases} (\frac{\|D_{vel}\|}{D_{cutoff}})^2 & \text{if } \|D_{vel}\| \leq D_{cutoff}; \\ \phi_{inf} & \text{if } otherwise. \end{cases} \tag{12}$$

where

$$D_{loc} = T_{loc,i} - M_{loc,j} \tag{13}$$

and

$$D_{vel} = T_{loc,i} + T_{vel,i} \times \Delta t - M_{loc,j} \tag{14}$$

The overall cost is a linear and normalized combination from both location and velocity evaluation, here we simply set $\alpha_1 = \alpha_2 = 0.5$. In the future, size and color information can be easily integrated as well. The weight can also be adaptive as in [17].

$$c_{i,j}^t = \alpha_1 f_{loc}(T_i, M_j) + \alpha_2 f_{vel}(T_i, M_j) \tag{15}$$

5 Clique Manager

5.1 The design

The addition of clique manager is is to assist target handover across rooms (cliques). Before this point, we presented clique-level visual tracker using visual fusing and motion correspondence. Each instance of clique-level tracker are designed to deal with targets within one clique, and thus have only local track labels. If we want to globally maintain target's identity across the house, we need a mechanism to convert and manage global labels, and that's the job for the house-wide clique manager.

Initially this can be easily considered as another visual fusing problem, however, there are some difference between the two and some unsolved issues:

Within individual clique there are significant overlapping for camera views (around 1/3), while in the inter-clique cases, only some of the cameras have overlapping at the edge of cliques. The overlapping relation is somewhat complicated.

We are dealing with a different set of vision abstraction (as shown in Fig. 1), thus different perception tasks: previously we need to segment blobs into targets for proper feature extraction while here our major concern is to associate targets from different cliques. Someone might argue that the whole house can be treated as a big clique so that it's possible to fuse all camera's blobs and followed by motion correspondence. However, we have not yet fully understood the scalability of those algorithms, and there might be serious performance issues. Also it seems this design violates the intuition of clique and house.

We found the work by Makris, et. al [25] and later work by Stauffer [36] interesting. They learn the trajectory pattern from individual views and try to understand the scene structure. This knowledge can potential help them handle identity handover better. At this moment, we have not got working code for the clique manager.

6 System Evaluation

Up to now, quite limited experimentations have been done because of time and human resource constraints, and we only have very rough and coarse understanding of how such system will fit into the particular environment such as Aware Home. A lot of further work is needed before we can comfortably deliver the system and even possibly deploy similar software and hardware to other environments.

We have designed several scenarios to evaluate the system performance. In most of the cases, we use the second floor kitchen area. All scenarios are captured in sync using four cameras (cameraB4 - camerB7), and each testing case is composed of four video files of the same length.

6.1 Synchronized capturing

The system is able to capture video in sync from multiple cameras/video sources and write to a series of numbered frames in JPEG/BMP format. In this case, we also need to perform a manual post-processing by assembling frames into video files using Matlab. All the testing cases are generated in this way.

6.2 Background subtraction

We've engineered an implementation for the algorithm in [14]. In general, the algorithm delivers very reasonable and constant performance for in-door scenes. Especially, the algorithm is good at dealing with shadow and highlighting, which is not uncommon in indoor scenarios.

It seems the background subtraction module has some difficulty in handling foreground merging. It's simply hard to find a comfortable compromise between foreground merging and background maintaining. We might need more experimentation work on this or work out other solutions. The readers are encouraged to refer to Wallflower paper [37] for performance evaluations and new ideas on background subtraction.

6.3 Clique-level visual tracker

Our clique-level visual tracking is composed of two parts: visual fusing and motion correspondence.

As in our previous work [45], we have a testing case (1200 frame at sampling of 5 fps) about up to 4 people walking in the working space, and our clique-level tracker deliver reasonable tracking performance in real-time.

Typical polygons are shown in Fig. 4 with each pixel in the figure represent a square grid of $2cm$. Even there might exists ambiguities in individual views, targets can be successfully segmented by fusing multiple views.

Figure 4: Typical polygons

Fig. 5 compares the ground truth with the output of visual fusing and the whole clique-level tracker. Compared with ground truth, there are still erroneous spikes in the visual fusing output, which get eliminated by delaying decision making on target exit (using the heuristic termination counter). This can be also viewed as a temporal filtering and the cost of doing so is the lagged responses for target exit.

Besides quite accurate target count, the visual tracker also generates real-time inference of targets' states such as

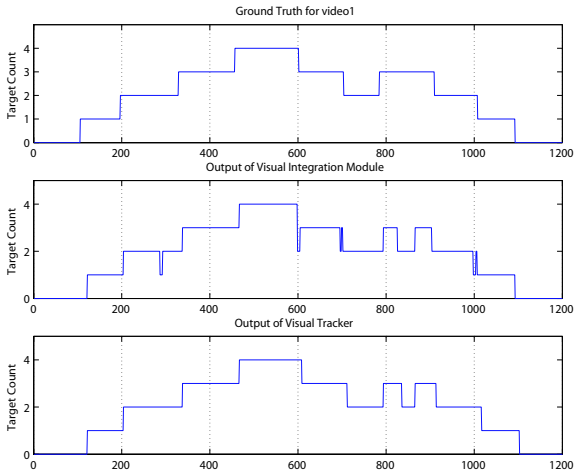

Figure 5: Target's count

location, velocity, and approximate size. For clarity, Fig. 6 and Fig. 7 shows X-t and Y-t plot of 3 out of 4 targets in the test case. Note the failure around frame 820, where the track 1(a) (green) and track 1(b) (black) are indeed for one target. The target stands very near to the wall, where the blob size is not big enough for cross-view validation, and thus target observation is lost for around 30 frames. The visual fusing algorithm did automatically recovered to correct counting. However, this interruption is a little bit longer to be justified as target recovery in motion correspondence algorithm. A new track is thus initiated, and previous track/identity is lost.

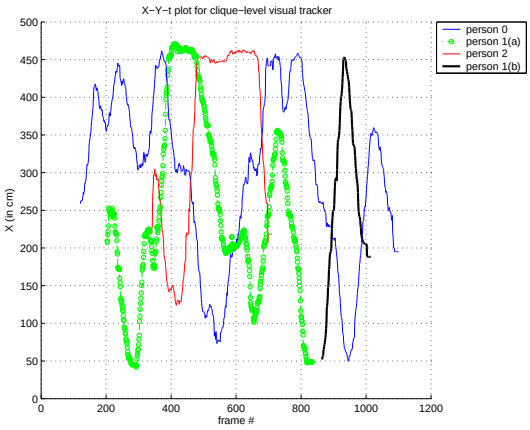

Figure 6: Target's trajectory: x-t

Also in [45], we have another test case of one person walking around the kitchen. This essentially removes the motion correspondence burden and can be used to evaluate the accuracy for visual tracking if ground truth trajectory is

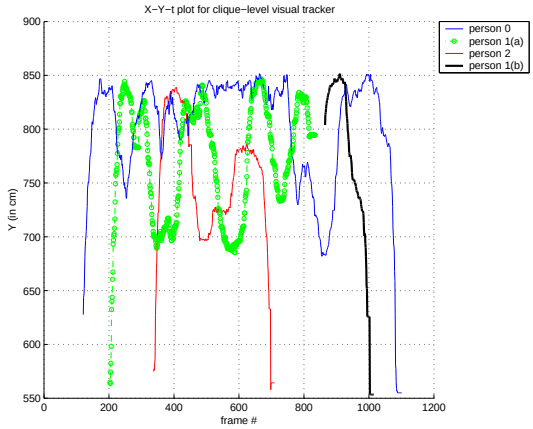

Figure 7: Target's trajectory: y-t

available.

6.4 Stress test

There is another 675-frame synchronized video set at the sampling rate of 5 fps. We design the experiment to be difficult by: 1) Up to four people enter the kitchen area dynamically; 2) When four people are all in the kitchen area, the density is considered relatively high since there is a dining table occupying 1/3 of the kitchen space that can not be used; 3) we especially design their interactions to be complex.

With this quite extreme scenario, we want to evaluate the system performance and get an understanding of where and why the system will fail.

This experimentation is inspired by [18] where the authors used the similar technique to evaluate a visual tracking in a smart room. Here is some design details for the experiment:

The time line for the videos in seconds: (see footnote for definitions)

0-20	four people come into the scene;
20-40	four pass-bys;
40-60	3-person in-and-out, followed by one large group merge;
60-90	deposit object, followed by two merge-and-stops and two merge-and-travels;
90-100	remove the object;
100-120	four people leave the scene.

The major problem is still the visual segmentation. Even our fusing algorithm uses multiple cameras for visual segmentation at its best efforts, in extreme case, such as large group merge, there is simply no way to segment and locate distinct targets. This will put extra burden on all subsequent algorithms. Because 1) to keep temporal smoothness, in visual fusing we tried to extrapolate from previous target locations when not sufficient observations can be made (remember the case where the polygon node being not tight). This causes potential issue, especially during target merging/split; 2) the motion correspondence complicated this issue by introducing heuristics on target enter/exit. The original algorithm [44] and our work [45] has tried with only limited success, and there are still a lot of design and improvement work needed to be done here.

6.5 Summary

Some observations from the experimentations:

- For shadows on the floor, even in individual views they tend to connect targets, visual fusing does a good job of segment them into targets: we are essentially able to get correctly segmented observation for each target in most cases;
- The reflection on the kitchen refrigerator door is sometime detected as blobs. However, it won't leak into further system since we explicitly model the 3D scene and the projection of those blobs lies outside valid tracking area;
- the clique-level visual tracker can perform reasonable well under moderate scenarios, especially when the number of target is smaller than 4 and without complicated interactions. Under extreme case, the limitation from hacks in visual fusing and motion correspondence becomes apparent and overall performance degrades;

7 Application

Essentially, this visual tracker can provide a real-time service of target's location, velocity (maybe other optional state) in the house. Various location-aware applications can be built upon this service.

[1] *Pass-by*: two people move near one another close enough, and pass each other; *in-and-out*: two people stand side by side, rapidly lean in and out so they merge and split again; *merge-and-stop*: two people walk towards each other to merge and then stop for several seconds before splitting again; *merge-and-travel*: two or more people merge, walk around the room very close together before split again; *large group merge*: three or more people merge together and splitting; *object deposit*: deposit object in the scene, and later remove it.

7.1 Finding lost objects

Given target's real-time location information, we can automatically direct the nearest camera to capture a snapshot of the target's activity. The concatenation of these snapshots will represent a specific target's activity over time. Even though accurate activity analysis is hard, this video can at least provide some visual cues for a human operator to figure out the past activity.

We cast this idea into the application of finding lost object: by visually inspecting those automatically captured snapshot videos, we help home residents to go through their activities and find their lost objects. Refer to [30], [38].

7.2 Acoustic map building

Robot can navigate a new environment based on acoustic information and perform certain tasks. In this sense, building acoustic map is part of the scene modeling process. In this application we use robot to automatically build the acoustic map, while vision system provide a real-time location feedback. See [9] for details.

8 Future directions

8.1 Alternative techniques

Joint Probabilistic Data-Association Filter (JPDAF) [10] is not suitable for our application since it's mainly used for situations with clutter.

Multiple Hypothesis Tracking (MHT) [31] [8] is known for its conceptual complexity to keep multiple hypothesis against temporal visual ambiguities. Actually Ingemar Cox published his implementation code for visual tracking (http://www.ee.ucl.ac.uk/ icox/). However, MHT is a statistical approach, and requires parameters that are not trivial to determine, and seems the algorithm is quite sensitive to the parameters, see [39] and [40]. The additional multi-camera setting further complicates the use of MHT in our application.

8.2 Condensation

Another important family of algorithms worth investigation is CONDENSATION (or similar particle filtering technique). Generally the problem is that even though condensation-like algorithms claim they can maintain multi-modality for the target distribution, they often confuse multi-modality with identity in multi-target tracking scenarios. Also it's just hard to tell whether a sudden modality is due to noise or alternatively due to a new target. For this reason, there is very few work on multi-target visual tracking using Condensation.

For multi-target tracking, condensation algorithms can run in a single target's state space vs. the multi-target joint space. Tracking in multi-target joint space is always challenged by the curse of dimension. As the number of target under tracking increasing, exponentially increasing number of particles will be needed to maintain tracking quality. This issue is further complicated by the dynamic number of targets as required in our settings. As a result, tracking in single-target space is very desirable over their joint-space tracking counterpart.

Through extensive literature reviewing, one practical solution is to de-compose the multi-target state space into multiple independent or nearly-independent single-target space so that the complexity is reduced dramatically to make the whole problem tractable. The basic assumption here is that targets only interact in small occasions, and for most of the time the multi-target posterior can be safely modeled as a set of independent single-target posteriors. But we still need a carefully-designed methodology to formalize this assumption. In addition, how to take advantage of multi cameras is another issue that has not even been fully addressed.

I'll comments on different approaches to multi-target visual tracking:

Distinguishable targets in single space: As in [41], the global posterior is modeled by a mixture of per-target PFs (particle filters). Each per-target PF is tuned and locked on one target, independently of other PFs. This approach precisely address the problematic feature of traditional particle filter that it fails to maintain consistent multi-modal distribution because of finite number of particles. The mixture component only interact when component weights are computed each step so that a global multi-modal posterior for the whole world can be constructed.

As part of the dynamic model, particles can be organized and re-clustered to handle addition/deletion of targets. The flip side of this approach is that it can not handle occlusion while at the same time maintaining consistent identity since it makes the decision instantaneously. With one object completely occluded, the naive re-clustering process would simply remove the target, losing all identity information previously associated with it. Also for this reason we argue that an occlusion-free observation model will greatly help the task of maintaining consistent identity information of targets in general multi-target tracking domain.

There is also an implementation paper [29] for this approach, where objects are detected and proposed using AdaBoost.

Indistinguishable targets in single space: [21] suggests the idea of tracking large number of targets using nearly-independent PFs, which is otherwise impossible in joint-space tracking because of the formidable number of particles needed. It also suggests an interesting idea of us-

ing MRF (Markovian Random Field) to penalize potentially confusing configurations under the assumption that intelligent targets under tracking will tend to avoid bumping into each other. With the use of MRF, the data association ambiguity (caused by the lack of distinguishable feature of targets) is dramatically reduced, so that simple dynamics can be sufficient to maintain multi-target tracking. In this sense, the role of MRF is equivalent to make individual target *identifiable* so that data association is generally simplified, which basically shares the same idea with [41]. The PFs are nearly independent in the sense that the MRF computation requires the knowledge of all target state hypothesis, where the *interaction* takes place. The flip side of using MRF is that we need to know the number of targets beforehand, since MRF is only meaningful given number of targets is known or hypothesized.

Another approach in this flavor is [35], which is a *dimensionally-compressed* version of [34] if we are willing to sacrifice targets' identity information. Assuming target indistinguishable, the author maintains the tracking in single-target space using multi-modal posterior. Also the algorithm is able to handle varying number of targets. However, as the approach suggests, we still need a post-processing to associate each peak (models) in the single-target multi-modal posterior with internal targets, using geometrical constraints or other distinguishing features, which also might be a source of ambiguity and error. However, this approach seems appealing because of its simplicity.

Indistinguishable targets in joint space: The Bramble [19] addresses the multi-target tracking problem by maintaining a posterior distribution in multi-target state space. By modeling both foreground and background, a multi-blob likelihood function enables direct weight comparison between particles with different dimensionality (containing different number of targets). The particle set, containing particles with different length, enables it to deal with dynamic object enter/exit, and provide a posterior estimate of number of targets each time step. The biggest issue for Bramble is that it works in the joint space to avoid explicit data association at the cost of exponentially exploding number of particles. The targets in Bramble are indistinguishable in the sense that no per-target foreground model is defined and maintained, which is the cause of possible mistakenly swapping identity especially during interaction and occlusion. The possibility of using per-target foreground model is discussed in their original paper; however, the result is discouraging since it suffered from the traditional problems associated with adaptive templates.

Distinguishable targets in joint space This constitutes the most complicated scenario of all cases. First, theoretically data association has to be explored considering every possible association between the permutations of targets and the permutations of observations. Second, the data association is further complicated if we explicitly model clutter (false positive) and missing targets (false negative) in observation model, because we are now dealing with un-equal number of models and observations and have to explain the difference. Third, when we are forced to consider dynamic target enter/exit, data association possibilities are further exploded since we are adding another factor to the un-equal number of models and observations. [34] summarized an approach exactly that way with huge time and space complexities, though.

8.3 Improvement on system design

Generally, the major consideration in designing a visual tracker is 1) how to prioritize application requirements for complex vision tasks; 2) how to gradually remove ambiguities and ensure self-recovery in face of ambiguity. The ambiguities may come from visual occlusion, background clutter, fluctuations/dramatic change in environment conditions (for example, illuminations) and even algorithm/sensor failures.

target modeling For data association, we have not yet fully integrated color model into motion correspondence, this might be another working area. Person modeling (shape/appearance) is always difficult at medium view range where significant perspective distortion exists. For appearance model, [27] uses normalized RGB for color-based matching across cameras in visual tracker.

Also we have not implemented part of [40] for global optimality, that might be another working area.

Integrate human factor For usability study, one potential research direction is to investigate how to include human factor on the fly into the motion correspondence to ensure long-term reliability for complex surveillance tasks.

Two-way communication For system-level design, currently all communication are one-directional from bottom to top. A potential research direction is to investigate how knowledge from top to bottom can help the system.

Scene learning As surveillance task shifts more and more from low-level motion detection to high-level of scene understanding, a cognitive vision system [5] that can "know", "understand" and "learn" the scene is highly desired. Possible research will be focused to build system that can learn the scene and automatically build a useable knowledge base with vast amount of tracking data, even possibly change the system's behavior adaptively. See Tim Ellis of City University London for his work on this area.

9 Summary and conclusions

The contribution of this paper are:

- Give design details about the whole visual tracker in Aware Home in order to gradually remove ambiguities;
- Conduct preliminary performance evaluation in Aware Home settings;
- Summarize for potential directions;

Acknowledgments

The support of resources from Aware Home Research Initiative (AHRI) at Georgia Institute of Technology to make my work possible is gratefully acknowledged.

Appendix

Appendix A: heuristics in the system

1. $T_{minSize}$: a heuristic value of the minimum size of a target (in grid count);
2. T_{thres}: threshold for confirmed targets within one polygon, suggest value 0.5;
3. ϕ_{thres}: the cost threshold to tell whether it's necessary to start a new track instead of associating measurement with a existing track;
4. N_{term}: a heuristic value determining how many frames we have to wait before terminating a track;
5. D_{cutoff}: cut-off distance for computing association cost;
6. α_1, α_2: fusing weight for distance and velocity into association cost;

References

[1] David Beymer, Kurt Konolige, Real-time tracking of multiple people using stereo, IEEE workshop on frame rate applications, methods, and experiences with regularly available technology and equipment, 1999.

[2] Aaron F. Bobick, Stephen S. Intille, James W. Davis, Freedom Baird, et. al., The KidsRoom: A perceptually-based interactive and immersive story environment, *Presence: Teleoperators and Virtual Environments*, August, 1999.

[3] Q. Cai, J. K. Aggarwal, Automatic tracking of human motion in indoor scenes across multiple synchronized video streams, *ICCV*, pp. 356-362, 1998.

[4] D. Caveney and J.K. Hedrick. Multiple Target Tracking in the Adaptive Cruise Control Environment Using Multiple Models and Probabilistic Data Association. *Proc. of ASME IMECE*, Design Engineering-23274, New York,NY, Nov. 2001.

[5] A. Cohn, D. Magee, A. Galata, D. Hogg, S. Hazarika, Towards an Architecture for Cognitive Vision using Qualative Spatio-Temporal Representations and Abduction, Proc. Spatial Cognition , June 2002.

[6] Collins, Lipton, Kanade, Fujiyoshi, Duggins, Tsin, et. al., A system for video surveillance and monitoring: VSAM final report. *Technical report CMU-RI-TR-00-12*, Robotics Insitute, Carnegie Mellon University, May, 2000.

[7] Robert Collins, Alan Lipton, Hironobu Fujiyoshi, Takeo Kanade, Algorithms for cooperative multisensor surveillance, *Proceedings of the IEEE*, Vol. 89, NO. 10, Oct. 2001.

[8] I. J. Cox, S. L. Hinorani, An efficient implementation of Reid's multiple hypothesis tracking algorithm and its evaluation for the purpose of visual tracking. *IEEE PAMI*, Vol. 18, No. 2, pp. 138-150, Feb., 1996.

[9] Eric Martinson, Ronald C. Arkin, Noise maps for acoustically sensitive nativation, Proc. of SPIE, Vol. 5609, Oct. 2004.

[10] T. E. Fortmann, Y. Bar-Shalom, M. Sheffe, Sonar tracking of multiple targets using joint probabilistic data association, *IEEE J. Oceanic Eng.*, Vol. 8, No. 3, pp. 173-184, July 1983.

[11] D. M. Gavrila, the analysis of human motion and its application for visual surveillance, Proc. of the 2nd IEEE International workshop on visual surveillance, pp. 3-5, Fort Collins, USA, 1999.

[12] I. Haritaoglu, D. Harwood, L. Davis, W4: Who, When, Where, What: A real time system for detecting and tracking people. *Third face and gesture recognition conference*, Pages: 222-227, 1998.

[13] I. Haritaoglu, D. Harwood, L. Davis, W4S: a real time system for detecting and tracking people in 2.5D, *European Conference on computer vision*, 1998.

[14] Thanarat Horprasert, David Harwood, Larry Davis, A statistical approach for real-time robust background subtraction and shadow detection, *ICCV'99 Frame-Rate Workshop*.

[15] Weiming Hu, Tieniu Tan, Liang Wang, Steve Maybank, A survey on visual surveillance of object motion

and behaviors, *IEEE Trans. on System, man and cybernetics*, Vol. 34, No. 3, Aug. 2004.

[16] Kohsia S. Huang, Mohan M. Trivedi, Video arrays for real-time tracking of person, head, and face in an intelligent room, Machine vision and applications (2002) 14: 103-111.

[17] Stephen S. Intille, Aaron F. Bobick, Closed-world tracking, *the 5th International Conference on Computer Vision*, pp. 672-678, June, 1995.

[18] Stephen Intille, James Davis, Aaron F. Bobick, Real-time closed-world tracking, *Computer Vision and Pattern Recognition*, pp. 697-703, Puerto Rico, June 1997.

[19] M. Isard, John MacCormick, Bramble: a Bayesian multiple-blob tracker, ICCV 2001

[20] Sumer Jabri, Zoran Duric, Harry Wechsler, et. al., Detection and Location of People in Video Images using Adaptive Fusion of color and edge information, *International Conference on Pattern Recognition*, Sep, 2000, Barcelona, Spain.

[21] Zia Khan, Tucker Balch, Frank Dellaert, Efficient particle filter-based tracking of multiple interacting targets using an MRF-based motion model, IROS 2003

[22] Zia Khan, Tucker Balch, Frand Dellaert, MCMC-based particle filtering for tracking a variable number of interacting targets, submit to PAMI, July 2004.

[23] J. Krumm, S. Harris, B. Meyers, B. Brumitt, M. Hale, S. Shafter, Multi-camera multi-person tracking for easyliving, *IEEE Inter. Workshop on visual surveillance*, 2000.

[24] Ruijiang Luo, Yan Guo. real-time stereo tracking of multiple moving heads, *ICCV workshop on recognition, analysis and tracking of faces and gestures in real-time systems*, 2001.

[25] D. Makris, T. J. Ellis, J. Black, Bridging the gaps between cameras, IEEE CVPR 2004, Washington, DC, June 2004.

[26] E. Mazor, A. Averbuch, Y. Bar-Shalom, J. Dayan, Interacting multiple model methods in target tracking: a survey.

[27] Anurag Mittal, Larry S. Davis, M2Tracker: A Multi-view Approach to Segmenting and Tracking People in a Cluttered Scene Using Region-Based Stereo, ECCV, 18-36, 2002.

[28] Rhys Newman, Yoshio Matsumoto, Sebastien Rougeaux, et. al., Real-time stereo tracking for head pose and gaze estimation, *4th IEEE International conference on automatic face and gesture recognition*, Grenoble, France, 2000.

[29] Kenji Okuma, Ali Taleghani, et. al. A boosted particle filter: multi-target detection and tracing, ECCV 2004

[30] Rodney E. Peters, Richard Pak, Gregory D. Abowd, Arthur D. Fisk, Wendy A. Rogers, Finding lost objects: informing the design of ubiquitous computing services for the home, GIT-GVU-04-01.

[31] D. B. Reid, An algorithm for tracking multiple targets, *IEEE Trans. Automatic Control*, Vol. 24, NO. 6, pp. 843-854, DEc., 1979.

[32] P. Remagnino, A. I. Shihab, G. A. Jones, Distributed intelligence for multi-camera visual surveillance, *Pattern Recognition*.

[33] I.K. Sethi, R. Jain, Finding trajectories of feature points in a monocular image sequence, IEEE Trans. PAMI, Vol. 9, No. 1, pp. 56-73, Jan,. 1987.

[34] Hedvig Sidenbladh, Sven-Lannart Wirkander, Particle filtering for random sets, IEEE Trans. on Aerospace and electrionic systems, March, 2003

[35] Hedvig Sidenbladh, Multi-target particle filtering for the probability hypothesis dentisy, 6th International conference on information fusion, 2003

[36] Chris Stauffer, Learning to Track Object Through Unobserved Regions, Proceedings of the IEEE Workshop on Motion, 2005.

[37] K. Toyama, J. Krumm, B. Brumitt and B. Meyers: Wallflower: Principles and practice of background maintenance, International Conference on Computer Vision (1999) pp. 255-261.

[38] Truon, Khai. N., Abowd, Gregory D., Brotherton Jason A. Who, what, when, where, how: Design issues of capture and access applications, Proceedings of the Inernational conference: Ubiquitous computing (UbiComp 2001), Atlanta, Georgia, Sep, 2001, pp. 209-224.

[39] Resolving motion correspondence for densely moving points, *IEEE Trans on PAMI*, Vol. 23, NO. 1, Jan. 2001.

[40] Establishing motion correspondence using extended temporal scope, *Artificial intelligence*, Vol. 145, issues 1-2, pp. 227-243, Apr. 2003.

[41] Jaco Vermaak, Arnaud Doucet, Patrick Perez, Maintaining multi-modality through mixture tracking, ICCV 2003

[42] Recent developments in human motion analysis, Liang Wang, Weiming Hu, Tieniu Tan, *Pattern Recognition*.

[43] Christopher Wren, Ali Azarbayejani, et. al., Pfinder: real-time tracking of the human body, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, July, 1997, vol. 19, No. 7, pp. 780-785.

[44] Danny Yang, Hector Gonzalez-Banos, Leonidas Guibas, Counting people in crowds with a real-time network of simple image sensors, *ICCV 2003*.

[45] Zhonghao Yang, Aaron Bobick, Visual integration from multiple cameras, *IEEE workshop on applications for computer vision, 2005*